



# Riego Inteligente

## HECHO CON ARDUINO

Por Irene Rodríguez García | Trabajo de Investigación de Bachillerato IES María Guerrero

## TABLA DE CONTENIDOS:

1. Introducción.....	2
2. Definición de Objetivos.....	2
3. Antecedentes previos. Estado de la Técnica.....	4
4. Metodología empleada.....	5
5. Diseño y realización.....	7
5.1 Modulo 1: Esquema de objetivos y elección de componentes.	
5.2 Modulo 2: Led y potenciómetro	
5.3 Modulo 3: LDR	
5.4 Modulo 4: DHT	
5.5 Modulo 5: IR	
5.6 Modulo 6: Display	
5.7 Modulo 7: Sensor de Humedad	
5.8 Modulo 8: Thermoresistor	
5.9 Modulo 9: Led de alarma	
5.10Modulo 10: Investigación de plantas	
6. Resultados obtenidos. Ensayos.....	36
7. Conclusiones, trabajo por hacer y mejoras.....	38
8. Documentación y Bibliografía.....	38
9. Anexos.....	41

## 1. Introducción

Un riego inteligente ( Smart Watering) es que toda la acción de regar la decidirá el dispositivo. El dispositivo inteligente no esta dirigido por el tiempo, él leera el ambiente y dependiendo de que tipo de planta es elegida si es más oportuno regar.

El Smart Watering es personalizado al tipo de cultivo. Según el cultivo seleccionado ( Olivo, Tomate, etc.) calcula el momento oportuno de regar según la cantidad de agua, temperatura y sol que haya recibido.

Un problema que atiza a la sociedad son el problema con el recurso de agua. Los problemas de abastecimiento de agua son constantes, llegan a pasar días o incluso semanas enteras sin salir agua de las llaves. Por ello, el objetivo principal de este proyecto es proteger al medio ambiente. Se consigue ahorra agua ya que solo riega cuando es necesario.

Otra característica es que es independiente del tiempo y zona geográfica. Sabe exactamente cuando regar; si hace frio, es mejor regar por el día ya que el agua puede congelarse por la noche y quemarla; si hace calor, es mejor regarla a por la mañana o cuando el sol haya caído.

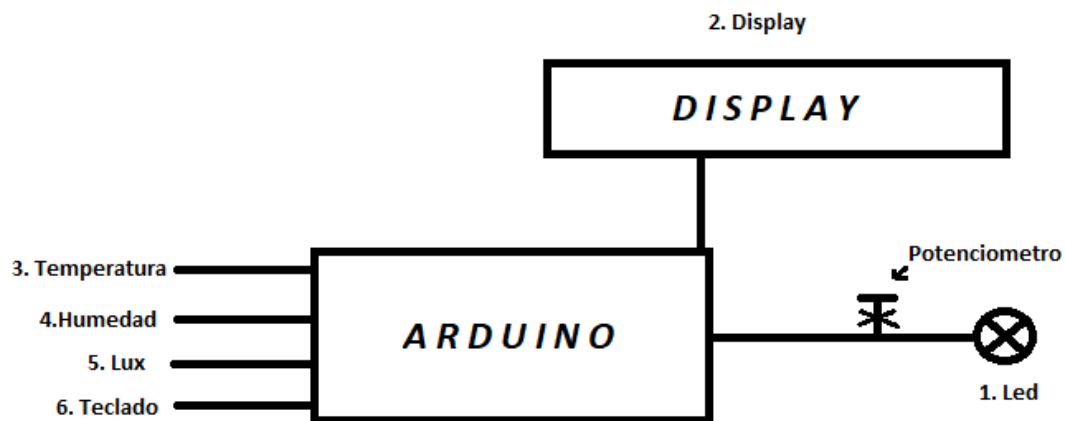
El riego inteligente esta orientado al sector agrícola con beneficios de un ahorro económico en el agua y un incremento de la productividad con un alto rendimiento.

## 2. Definición de Objetivos:

El trabajo de investigación tiene como **principal objetivo** crear un sistema de control inteligente o automatico. Es decir tiene la habilidad para actuar de forma apropiada en un entorno incierto. Para el cual usando piezas leegoo y arduino se ha construido un dispositivo de riego capaz de analizar, organizar y convertir los datos en información estructurada.

El sistema tiene como principales elementos la placa arduino para crear los elementos autónomos, conctándose a los dispositivos e interactuar con el Software y el Hardware.

En el Hardware los principales elementos son un sensor de humedad de tierra, un sensor de temperatura y un fotoresistor (LDR) para medir la luz. En vez de utilizar una tobera se ha implantado un led que se encendera al regar y se añadira un potenciómetro para poder elegir la fuerza con la que sale el agua. Para poder mirar los valores se añadira un display que enseñe los valores de cada sensor.



*Fig. 1 – Esquema del Hardware 10/06/2019*

Una vez hecho los objetivos del Hardware pasamos a los programas:

- I. Potenciometro más el led.
- II. Display.
- III. Temperatura, leer grados y ponerlo en el display.
- IV. Humedad leerla en porcentaje y mostrarla en el display.
- V. Lux y ponerlos en el display.
- VI. Teclado, elegir las plantas y mostrar en el display el tipo que se ha elegido.
- VII. Riego inteligente que es juntar todos los programas anteriores.

Durante la realización del proyecto también se han creado nuevos objetivos, los objetivos secundarios han sido:

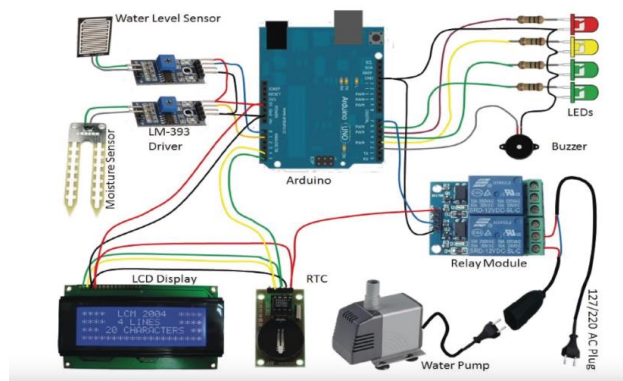
- Aprender nuevos comandos y lenguajes como el C++ para simplificar el programa.
- Experimentación con los elementos de Hardware por separado para saber como funcionan.
- Estudiar cuales son los valores optimos de temperatura, humedad y luz para regar.

Aunque el objetivo principal es lograr que el sistema resultante funcione, otro objetivo que se debe remarcar es usar todos los conocimientos adquiridos y aprender a enlazar la información obtenida.

### 3. Antecedentes previos. Estado de la Técnica

En este apartado se estudiara otros riegos que se asemejen a los del objetivo, donde se cojera influencia.

Se estudián proyectos parecidos para ver la competencia y comparar esquemas. Para así llegar a mejorar el producto.



*Fig. 2 – Influencia del riego inteligente URB.*

Comparando los demás proyectos, todos tenían algo en común tenían un led o un zumbador de alarma que serviría para avisar si ha pasado cierto tiempo sin regar, por ello se implento al final un led de emergencia en el esquema.

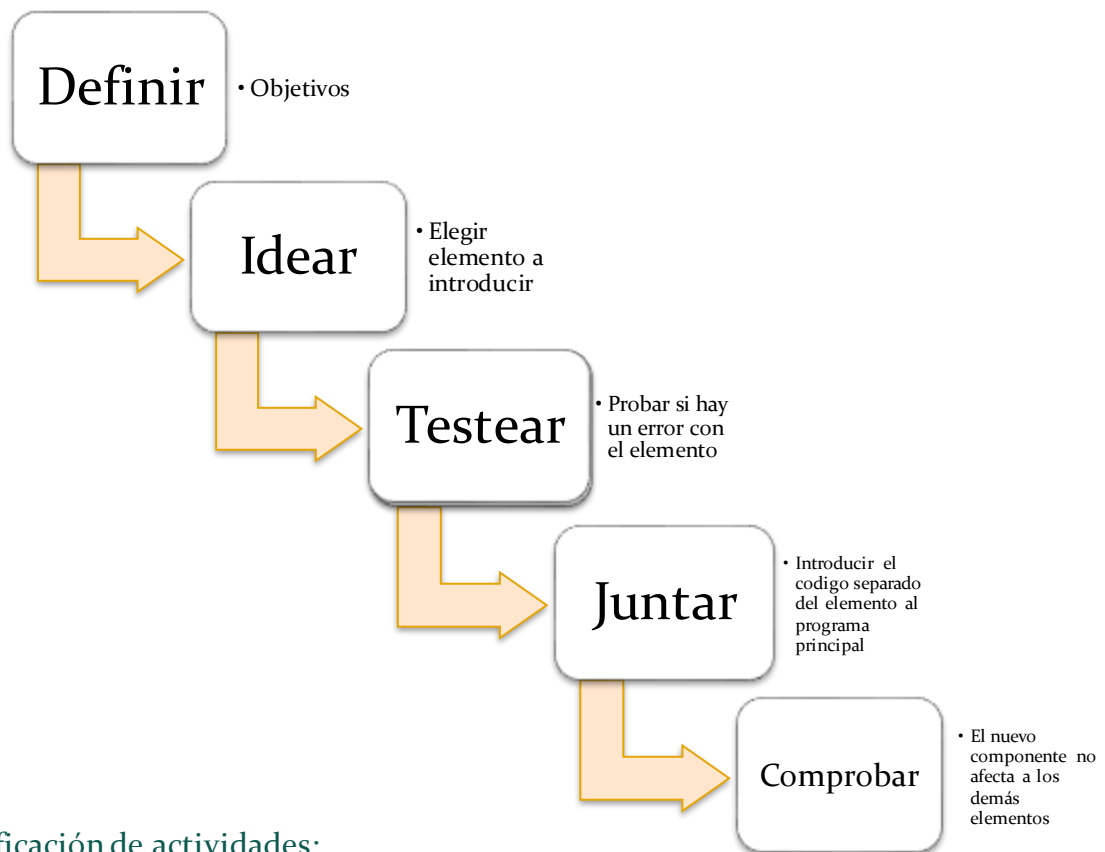
El futuro de la agricultura de regadío a nivel mundial depende, en buena parte, de la implantación de sistemas de riego inteligente en las fincas de cultivo, que permitan la utilización más eficiente de los recursos agua, fertilizante y energía de manera que se aumenten los niveles de producción utilizando menos recursos productivos. Debido a esto cada vez más empresas privadas intentan atacar con el objetivo de ahorrar agua y ser un riego optimo. Los elementos optimos para esto son según smart water summit un evento que se centra en los mismos objetivos del proyecto son:

- Sensores de Humedad del Suelo.
- Caudalímetros. Monitorización del caudal y volumen aplicado en el riego.
- Logger. Sistema de adquisición de datosel cual se conectán los equipos.
- Software de gestión del riego
- Sensores de temperatura
- Sondas de succión. Controlar la solución nutritiva del suelo
- Drones. Toma de decisiones del riego.

#### 4. Metodología empleada:

La metodología empleada ha sido definir, idear, testear, juntar y probar. Es decir:

- **Definir:** Primero se proponen los objetivos.
- **Idear:** Pensamos los elementos que puedan llevar a cabo las funciones necesarias para llegar al objetivo.
- **Testear:** Una vez se ha elegido el elemento por separado se prueba para comprobar si funciona y que comandos son los que llevan a cabo cada acción.
- **Juntar:** Una vez ya testeado el componente se introduce al programa principal que es el del riego inteligente.
- **Comprobar:** Que al integrarlo funciona correctamente y no afecta a los demás componentes.



#### Planificación de actividades:

Seguiendo esta metodología, para hacer el proyecto se ha buscado y recogido información de internet, libros y de profesionales. Seleccionada la información se pudo utilizar nuevos elementos, generar un código basado en otros existentes, aprendizaje de nuevos lenguajes, sobre todo C++ y arduino. Las etapas fueron las siguientes:

- Montaje del diseño del Hardware y análisis del esquema del programa principal.
- Probar individualmente cada elemento y uno por uno, añadirlos en el programa.
- Probarlos ver los fallos y cambiar elementos por otros.

- Cambiar los comandos a formas más simplificadas y elegantes usadas en C+.
- Probar el programa corrigiendo y ajustando los errores.

### Recursos:

Información Los recursos han sido documentación técnica y manuales de características (listados en la bibliografía), los componentes de Eleegoo y Arduino ( Placa de Eleegoo, potenciómetros, leds, thermoresistor, resistencias, LDR...) y eléctricos ( cables, batería...).

Para la programación se ha empleado librería de código abierto y gratuito, descargable en internet (Arduino). Se ha programado el código desde un ordenador personal.

Tabla de objetos utilizados:

COMPONENTES	CUANTOS UTILIZADOS
Placa arduino	1
Cables	41
Resistencias	4
Sensor de Humedad	1
Potenciometro	2
Led	2
IR	1
Sensor de Temp (Thermo-resistor)	1
Display	1
Photo-resistor (LDR)	1
Protoboard	3
Ordenador personal	1
Cable de alimentación	1
Bateria	1

El código final y los elementos finales usados estara explicado en el apartado 6 de la memoria.

## 5. Realización:

### 5.1 *Modulo 1: Esquema de objetivos y elección de los componentes.*

Cada planta necesita una cantidad de agua y no todas tienen los mismos tiempos ideales para regar, por ello se cogen 9 tipos de plantas y se determinan su temperatura, humedad y luz ideal para regar, si concuerda todo con los parámetros establecidos, este dispositivo regara, es decir el led de color verde se encenderá. Para elegir el tipo de planta utilizaremos un mando que funciona por infrarrojos el cual añadiremos al esquema.

Añadimos un display para que indique la planta que se ha elegido y los datos que reciben el sensor de humedad, temperatura y luz.

- Temperatura saldrá en grados
- Humedad en porcentaje
- Luz en lux

También añadiremos un potenciómetro para regular manualmente la intensidad con la que sale el agua por si por ejemplo el riego instalado deja salir más agua de lo previsto. Además, se implementará un led de color rojo que se encenderá en caso de alarma.

En total serían 7 programas que en el séptimo se unirían todos los anteriores haciendo el definitivo.

Todos los programas base han sido sacados de Elegoo. Fórmulas como la formula Steinhart-Hart van con sus respectivos creadores.

### 5.2 *Modulo 2: Led y potenciómetro.*

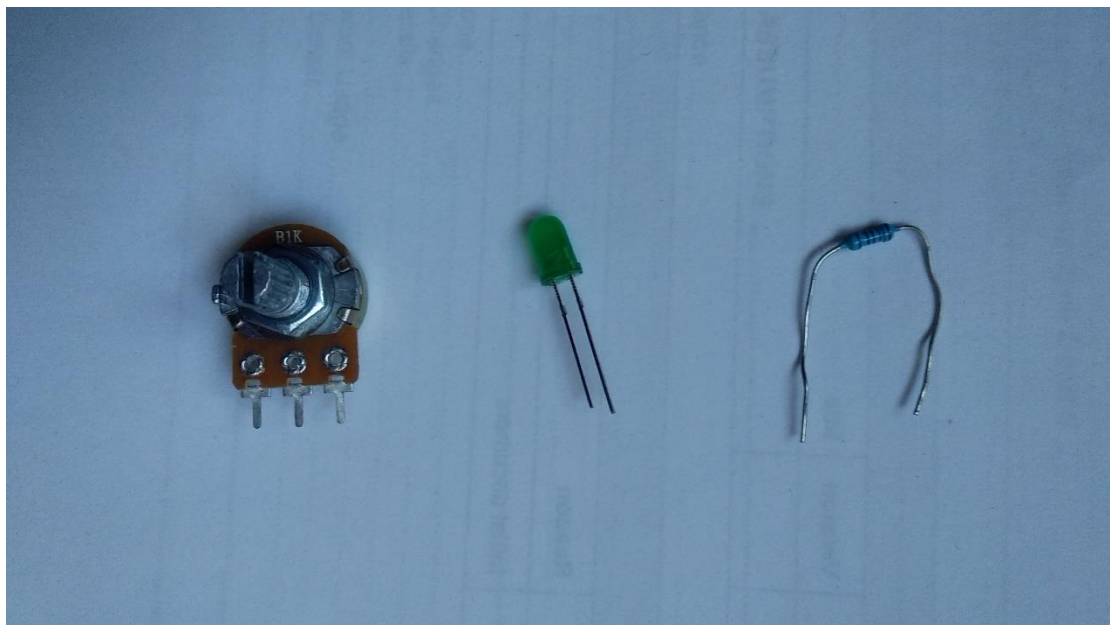


Fig. 3 – De izquierda a derecha, potenciómetro, led y resistencia.



. Primero hacemos el simulador de riego que será un led, cuando se encienda significará que está echando agua y el potenciómetro controlará la intensidad.

Primero comprobar que el led funcionaba con el código siguiente y utilizando una resistencia de 330 ohmios para no dañar el led. El led verde ira a la digital 3

```
int led = 3; // Variable

void setup() {

  Serial.begin(9600); //iniciar puerto serie.

  pinMode(led , OUTPUT); //definir pin como salida.

}

void loop(){

  digitalWrite(led , HIGH);
```

El led se encendió así que se comprueba que funciona. Lo siguiente es cambiar la resistencia por el potenciómetro. Quedando el Código:

```
int led = 3;  <-Definición del led

int sensorPin = A0;  <- Definición del potenciómetro

// Funciones del led, pin 3

void init_led(){      <- Es el setup del led

  pinMode(led, OUTPUT);

  Serial.begin(9600);

}

void turn_up_led(){  <- Función para encender el led

  // 0 .. 256

  int sensorValue = 0;

  int fade = 0;

  //

  sensorValue = analogRead(sensorPin);

  Serial.println(sensorValue);

  fade=map(sensorValue,0,1023,0,255);  <- Como los valores son analógicos 0-1023 hacemos un
mapeo para que vayan de 0 a 255

  Serial.println(fade);  <- Poder ver el valor del potenciómetro en el serial monitor

  analogWrite(led,fade);
```

```

}

void turn_down_led(){    <- para apagar el led

    analogWrite(led,o);

}

void setup() {

    init_led();

}

void loop() {

    turn_up_led();

    delay (500);

    turn_down_led();

    delay(500);

    // hace que se encienda por 5 segundos y se apague por 5 segundos constantemente debido al
    loop.

}

```

Ya está el código del led y del potenciómetro.

### 5.3\_Modulo 3: LDR.

La fotocélula utilizada es de un tipo llamado un resistor dependiente de la luz, a veces llamado ldr. Como su nombre indica, este componente actúa como una resistencia, excepto que la resistencia cambia en respuesta a cuanta luz está cayendo sobre él. El ldr mide la luz en lux.

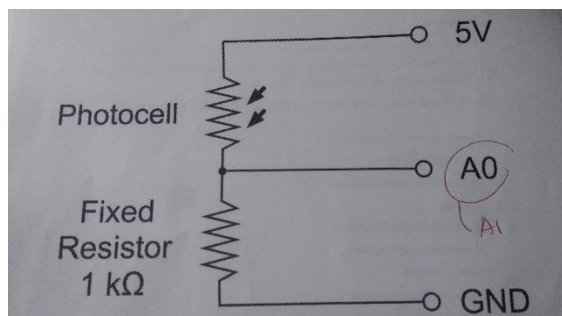


Fig. 5 – LDR Hardware.

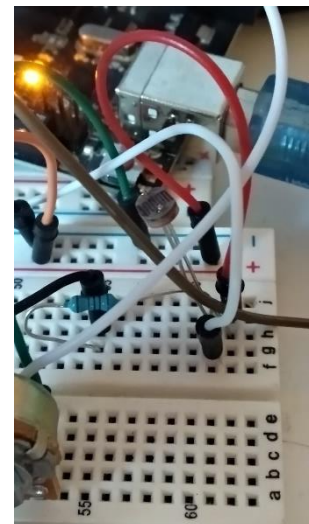


Fig. 4 – LDR y resistencia de 330 ohmios.

Como se explica en la imagen 5 el LDR se conecta a 5V (alimentación) y en vez de a A0 se cambia a A1 debido a que A0 lo está utilizando el potenciómetro del riego.

Ahora se prueba que el LDR funciona correctamente, el funcionamiento de este se aprendió a manejar en el instituto.

Primero las variables:

```
int LDRPin = A1;

// Valor de la Luz 0...100%

int valorLDR;

void LDR_Read(){  <- Función que lee el Ldr y lo enseña en el monitor serie,

    int fade;

    fade= analogRead(LDRPin);

    valorLDR=map(fade,0,1023,0,100);

//100 klx - 100.000 lux máx de unidad de medida de la luz

// Hacemos un mapeo de los valores analógicos a un porcentaje

    Serial.println(valorLDR);

    //

}
```

Añadiéndolo en el código principal donde esta el led y el potenciómetro:

```
led_pot_LDR
// definiciones del led
int ledPin = 3;
int PotentPin = A0;
int LDRPin = A1;

// Valor de la Luz 0..100%
int valorLDR;

void setup() {
    init_led();
}

void loop() {
    //
    LDR_Read();
    //
    turn_up_led();
    //
    delay (500);
    //turn_down_led();
    //delay(500);
}

// Funciones del led, pin 3

void init_led(){
    //
    pinMode(ledPin, OUTPUT);
    //
}
```

```

    Serial.begin(9600);
}

void turn_up_led() {
    // 0 .. 255
    int sensorValue = 0;
    int fade = 0;
    //
    sensorValue = analogRead(PotentPin);
    //Serial.println(sensorValue);
    fade=map(sensorValue,0,1023,0,255);
    //Serial.println(fade);
    analogWrite(ledPin,fade);
}

void turn_down_led() {
    analogWrite(ledPin,0);
}

void LDR_Read() {
    int fade;
    fade= analogRead(LDRPin);
    valorLDR=map(fade,0,1023,0,100);
    //
    Serial.println(valorLDR);
    //
}

//100 klx - 100.000 lux máx de unidad de medida de la luz

```

Fig. 6 – Código del LDR incluyéndolo en el código.

El código lo que hace es encender el led y controlar la intensidad con el potenciómetro, además de poner los valores que lee el LDR y la intensidad del potenciómetro en el monitor serie.

## 5.4\_Modulo 4: Temperatura y Humedad.

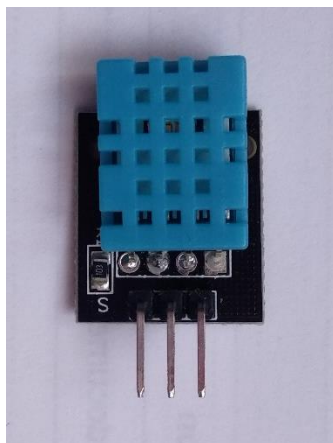


Fig. 7 – DHT11..

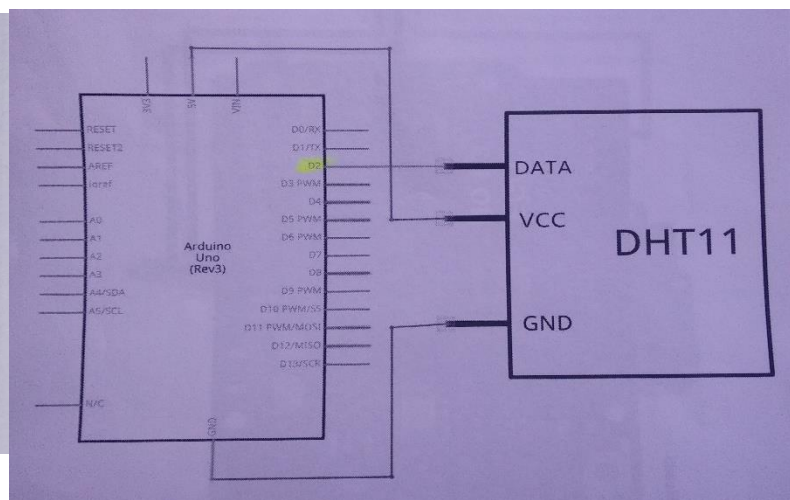


Fig. 8 – DHT11 Hardware, - Data a digital 2.

El sensor de temperatura y de humedad ambiental. Montamos el DHT11 siguiendo las instrucciones de eleegoo. El data va en Digital 2, Vcc a 5 voltios y Gnd a tierra.

En el programa:

```
DHT_nonblocking  dht_nonblocking.cpp  dht_nonblocking.h

//#include <dht_nonblocking.h>

#include "C:\Users\irene\OneDrive\Documents\proyecto_riego\dht\DHT_nonblocking\dht_nonblocking.h"

/* Uncomment according to your sensortype. */
#define DHT_SENSOR_TYPE DHT_TYPE_11
```

Se utiliza el #include para meter otros programas descargados en el programa.

El programa es sacado de Eleegoo.

Probándolo solo queda como el programa como:

```
DHT_nonblocking  dht_nonblocking.cpp  dht_nonblocking.h

/*
 * Poll for a measurement, keeping the state machine alive. Returns
 * true if a measurement is available.
 */
static bool measure_environment( float *temperature, float *humidity )
{
    static unsigned long measurement_timestamp = millis( );

    /* Measure once every four seconds. */
    if( millis( ) - measurement_timestamp > 3000ul )
    {
        if( dht_sensor.measure( temperature, humidity ) == true )
        {
            measurement_timestamp = millis( );
            return( true );
        }
    }

    return( false );
}

/*
 * Main program loop.
 */
void loop( )
{
    float temperature;
    float humidity;
```

```

/* Measure temperature and humidity.  If the functions returns
   true, then a measurement is available. */
if( measure_environment( &temperature, &humidity ) == true )
{
    Serial.print( "T = " );
    Serial.print( temperature, 1 );
    Serial.print( " deg. C, H = " );
    Serial.print( humidity, 1 );
    Serial.println( "%" );
}
}







```

---

Fig. 8.2 – DHT11 Software solo.

Este código hace que cada 3 segundos lea el dht11 y ponga los valores de temperatura en grados y la humedad en porcentaje en el monitor serie.

Como funciona seguimos la metodología y lo añadimos al código principal:

 dht_nonblocking.cpp	09/12/2016 2:18	Archivo CPP	8 KB
 dht_nonblocking.h	09/12/2016 2:18	Archivo H	2 KB
 IRremote.cpp	16/11/2014 6:20	Archivo CPP	32 KB
 IRremote.h	16/11/2014 6:20	Archivo H	5 KB
 IRremoteInt.h	16/11/2014 6:20	Archivo H	15 KB
 led_pot_LDR_DHT.ino	22/06/2019 19:23	Arduino file	4 KB

Primero en la carpeta donde está el código hay que añadir los informes subrayados para que el programa pueda encontrarlos y utilizarlos.

```

void loop() {
    //
    //Choose kind
    //IR_Read();
    //
    // Read photo resistor LUX
    LDR_Read();
    //
    // Read temp + humedad
    DHT_Read();
    //

    //
    // Enciende Led
    turn_up_led();
    //
    //delay (500);
    //turn_down_led();
    //delay(500);
}

```

En la función DHT esta exactamente lo mismo que el código del principio y hace lo mismo, pero armónicamente con los demás.

### 5.5\_Modulo 5: IR.

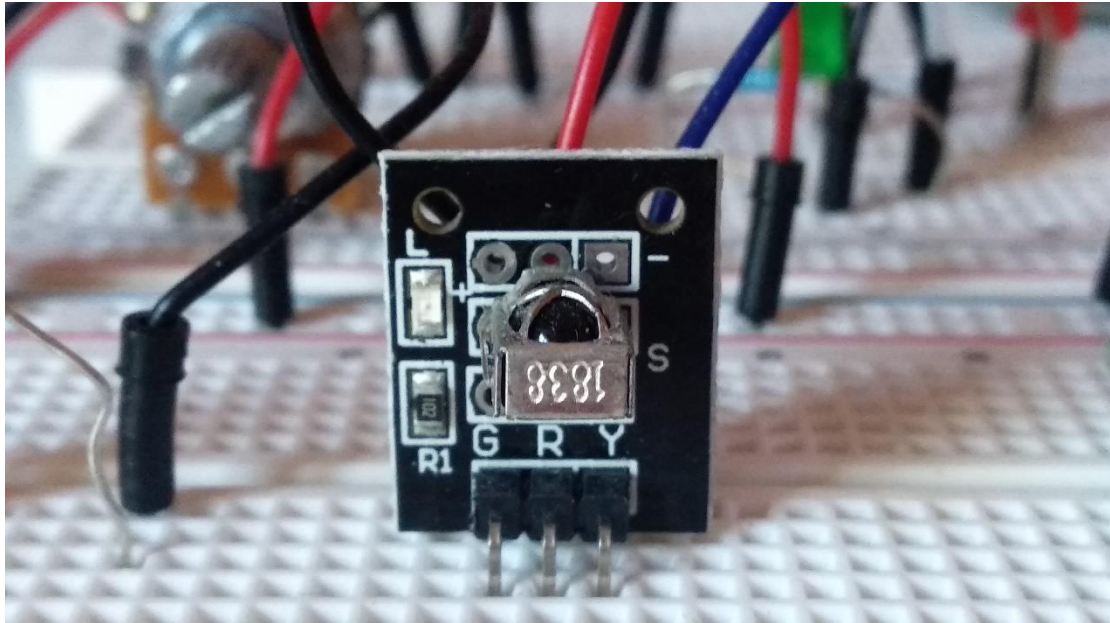


Fig. 9 – IR

La conexión es la de la figura 9.1. Como el digital 6 se ha reservado para el display, lo cambiamos para el digital 3 cuando se vaya añadir al código principal.

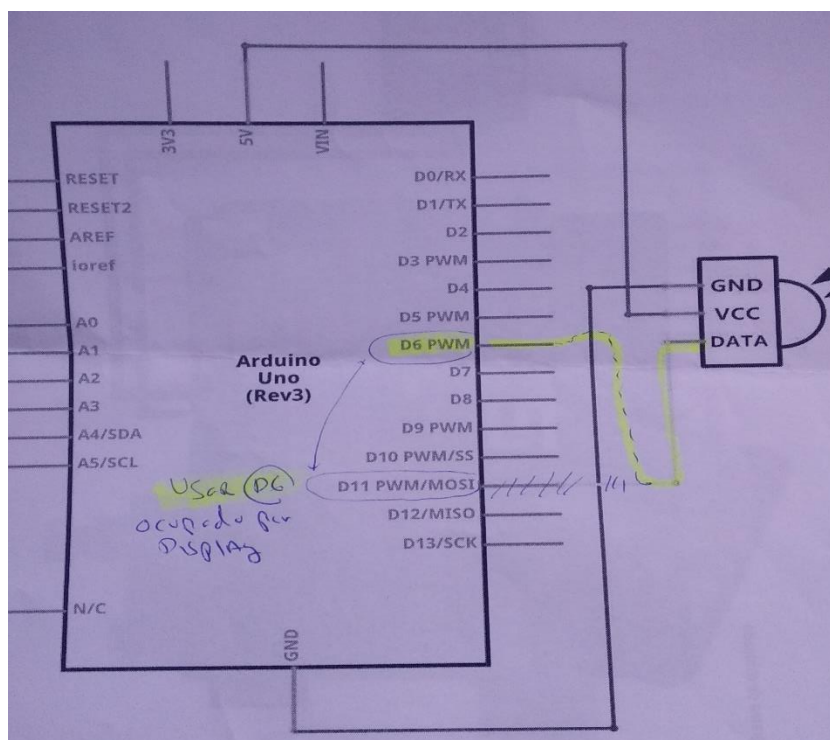


Fig. 9.1 – Conexiones del IR



Los detectores de infrarrojos son pequeños microchips con una célula fotoeléctrica que están configurados para recibir a la luz infrarroja. Casi siempre se utilizan para la detección de control remoto, en este caso se utiliza un mando. Con el mando elegimos el tipo de planta pulsando un número del 1 al 9.

Como se hizo con el infrarrojo necesita también sus informes.

Fig. 10 – Mando.

IR_Receiver_Module.ino	13/06/2019 18:19	Arduino file	3 KB
IRremote.cpp	16/11/2014 6:20	Archivo CPP	32 KB
IRremote.h	16/11/2014 6:20	Archivo H	5 KB
IRremoteInt.h	16/11/2014 6:20	Archivo H	15 KB

Infrarrojo Primero probamos que funciona los infrarrojos y vemos que funciona cada cosa. Lo que hace cada línea del código esta comentada al lado.

```

IR_Receiver_Module$ IRremote.cpp IRremote.h IRremoteInt.h

//www.elegoo.com
//2016.12.9

#include "IRremote.h"

int receiver = 6; // Signal Pin of IR receiver to Arduino Digital Pin 6

/*-----( Declare objects )-----*/
IRrecv irrecv(receiver);    // create instance of 'irrecv'
decode_results results;      // create instance of 'decode_results'

/*-----( Function )-----*/
void translateIR() // tiene acción depende del Ir recibido

// describing Remote IR codes

{

    switch(results.value)

    {
        //case 0xFFA25D: Serial.println("POWER"); break;
        //case 0xFFE21D: Serial.println("FUNC/STOP"); break;
        //case 0xFF629D: Serial.println("VOL+"); break;
        //case 0xFF22DD: Serial.println("FAST BACK"); break;
        //case 0xFF02FD: Serial.println("PAUSE"); break;
        //case 0xFFC23D: Serial.println("FAST FORWARD"); break;
        //case 0xFFE01F: Serial.println("DOWN"); break;
        //case 0xFFA857: Serial.println("VOL-"); break;
    }
}

```



```

//case 0xFF906F: Serial.println("UP");    break;
//case 0xFF9867: Serial.println("EQ");    break;
//case 0xFFB04F: Serial.println("ST/REPT"); break;    Como no nos interesa los demas
//case 0xFF6897: Serial.println("0");    break;    botones tan solo conservamos los de las plantas
case 0xFF30CF: Serial.println("1");    break;
case 0xFF18E7: Serial.println("2");    break;
case 0xFF7A85: Serial.println("3");    break;
case 0xFF10EF: Serial.println("4");    break;
case 0xFF38C7: Serial.println("5");    break;
case 0xFF5AA5: Serial.println("6");    break;
case 0xFF42BD: Serial.println("7");    break;
case 0xFF4AB5: Serial.println("8");    break;
case 0xFF52AD: Serial.println("9");    break;
//case 0xFFFFFFFF: Serial.println(" REPEAT");break;

default:
    Serial.println(" other ");
}
// End Case

delay(500); // No repitas inmediatamente espera 5 segundos

} //END translateIR
void setup()    /*----( SETUP: RUNS ONCE )----*/
{
    Serial.begin(9600);
    Serial.println("IR Receiver Button Decode");
    irrecv.enableIRIn(); // Empieza el receptor

}/*--(end setup )---*/

void loop()    /*----( LOOP: RUNS CONSTANTLY )----*/
{
    if (irrecv.decode(&results)) // ¿Hemos recibido la señal?
    {
        translateIR();
        irrecv.resume(); //recibe el siguiente valor
    }
}/* --(end main loop )-- */

```

Este código hace que en el monitor serie aparezcan los números pulsados del 1-9.

Como funciona lo incluimos en el código principal:

 dht_nonblocking.cpp		09/12/2016 2:18	Archivo CPP
 dht_nonblocking.h		09/12/2016 2:18	Archivo H
 IRremote.cpp		16/11/2014 6:20	Archivo CPP
 IRremote.h		16/11/2014 6:20	Archivo H
 IRremoteInt.h		16/11/2014 6:20	Archivo H
 led_pot_LDR_DHT.ino		22/06/2019 19:23	Arduino file

Añadimos los ficheros necesarios.

led_pot_LDR_DHT	IRremote.cpp	IRremote.h	IRremoteIn.h	dht_nonblocking.cpp	dht_nonblocking.h
-----------------	--------------	------------	--------------	---------------------	-------------------

```

#include "IRremote.h"
#include "dht_nonblocking.h"
#define DHT_SENSOR_TYPE DHT_TYPE_11

static const int DHT_SENSOR_PIN = 2;
DHT_nonblocking dht_sensor( DHT_SENSOR_PIN, DHT_SENSOR_TYPE );

// definiciones del led
int ledPin = 3;
int PotentPin = A0;
int LDRPin = A1;
// Valor de la Luz 0..100%
int valorLDR;
// definición del ir
int receiver = 6;
IRrecv irrecv(receiver);
decode_results results;

void setup() {
  init_led();
  //Serial.print( "hi" );
  tipodeplanta ();
}

void loop() {
  //
  //Choose kind
  //IR_Read();
  //
  // Read photo resistor LUX
  LDR_Read();
  //
  // Read temp + humedad
  DHT_Read();
  //

  //
  // Enciende Led
  turn_up_led();
  //
  //delay (500);
  //turn_down_led();
  //delay(500);
}

// Funciones del led, pin 3

```

```

void init_led(){
    //
    pinMode(ledPin, OUTPUT);
    //
    Serial.begin(9600);
}

void turn_up_led(){
    // 0 .. 256
    int sensorValue = 0;
    int fade = 0;
    //
    sensorValue = analogRead(PotentPin);
    //Serial.println(sensorValue);
    fade=map(sensorValue,0,1023,0,255);
    //Serial.println(fade);
    analogWrite(ledPin,fade);
}

void turn_down_led(){
    analogWrite(ledPin,0);
}

void LDR_Read(){
    int fade;
    fade= analogRead(LDRPin);
    valorLDR=map(fade,0,1023,0,100);
    //
    // Serial.println(valorLDR);

    //
}

//100 klx - 100.000 lux máx de unidad de medida de la luz

void DHT_Read() {
    float temperature;
    float humidity;
    if( measure_environment( &temperature, &humidity ) == true ) {
        Serial.print( "T = " );
        Serial.print( temperature, 1 );
        Serial.print( " deg. C, H = " );
        Serial.print( humidity, 1 );
        Serial.println( "%" );
    }
}

static bool measure_environment( float *temperature, float *humidity )
{
    static unsigned long measurement_timestamp = millis( );

    /* Measure once every four seconds. */
    //if( millis( ) - measurement_timestamp > 3000ul )
    if( millis( ) - measurement_timestamp > 1500ul )
    {

        if( dht_sensor.measure( temperature, humidity ) == true )

```

```

    {
        measurement_timestamp = millis( );
        return( true );
    }
}

return( false );
}

void tipodeplanta () {
    //Serial.begin(9600);
    Serial.println(" ¿Tipo de planta ( 1-9 )?");
    irrecv.enableIRIn();

}

void translateIR() {
    decode_results results;
    switch(results.value) {
        case 0xFF30CF: Serial.println("1");    break;
        case 0xFF18E7: Serial.println("2");    break;
        case 0xFF7A85: Serial.println("3");    break;
        case 0xFF10EF: Serial.println("4");    break;
        case 0xFF38C7: Serial.println("5");    break;
        case 0xFF5AA5: Serial.println("6");    break;
        case 0xFF42BD: Serial.println("7");    break;
        case 0xFF4AB5: Serial.println("8");    break;
        case 0xFF52AD: Serial.println("9");    break;
        default:
            Serial.println(" other ");
    }
    delay(500);
}

void IR_Read() {
    IRrecv irrecv(receiver);
    decode_results results;
    if (irrecv.decode(&results)) {
        translateIR();
        irrecv.resume();
    }
}
}

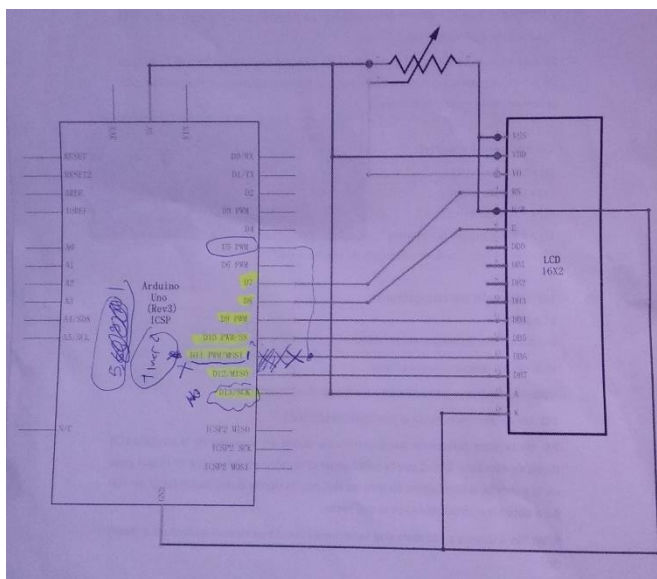
```

El código hace lo siguiente: Al inicializarlo en el monitor serie pregunta el tipo de planta, con el mando se elige la planta del 1 al 9 y luego empieza a leer valores de los elementos anteriores.

## 5.6\_Modulo 6: Display.



Fig. 11 – Display.



La pantalla tiene una retroiluminación de LED y puede mostrar dos filas con hasta 16 caracteres en cada fila. La pantalla es blanca en azul, diseñada para mostrar texto

Para montarlo se necesita un potenciómetro de 10k que ajuste el brillo del display. Las conexiones son:

VSS: A tierra

VDD: a +5V la fuente de alimentación

Fig. 11 – Display conexiones.

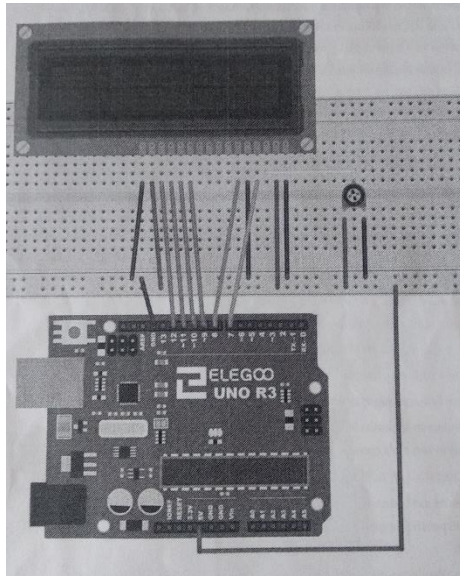
VO: Un pasador que ajusta el contraste del display

RS: Pin que controla donde en memoria de la pantalla LCD datos de escritura.

R/W: Pin A lectura y escritura.

Las demás conexiones son las mostradas en la figura 11 y en la de abajo.

Fig. 11.2 – Display conexiones.



En cuanto al código primero se comprueba que funciona el display.

Queremos utilizar la biblioteca de cristal líquido.

Display.ino	26/06/2019 20:59	Arduino file	3 KB
LiquidCrystal.cpp	22/11/2016 22:05	Archivo CPP	9 KB
LiquidCrystal.h	22/11/2016 22:05	Archivo H	3 KB

Tras probar el código y aprender la forma que funciona el código el cual esta explicado al lado de cada línea comentado (código de abajo), hay que organizar el espacio del lcd 2 x 16.

Código de prueba:

```
// incluir el código de la biblioteca:
#include "LiquidCrystal.h"

//Estos son los pines del display están en la conexión

LiquidCrystal lcd(7, 8, 9, 10, 5, 12);

void setup() {
  //Esto funciona para decir desde donde empieza a escribir
  lcd.begin (16,2);

  // Con el comando lcd.print("") podemos enviar a que ponga esto
  //Como esta en el SetUp solo se mostrara una vez
  //lcd.print("hello world...");
}
```

Ahora que se ha comprobado que el display funciona correctamente organizamos el espacio:

El primer diseño fue:

En setup->

i	T	i	p	o		d	e		p	l	a	n	t	a	?

En loop->

p	l	a	n	t	a			r	e	g	a	n	d	O	
3	2	°			4	o	%		5	o	l	X			

El diseño final elegido:

Setup:

<i>T</i>	<i>i</i>	<i>p</i>	<i>o</i>		<i>d</i>	<i>e</i>		<i>p</i>	<i>l</i>	<i>a</i>	<i>n</i>	<i>t</i>	<i>a</i>	<i>?</i>	
<i>E</i>	<i>l</i>	<i>e</i>	<i>g</i>	<i>i</i>	<i>r</i>		<i>e</i>	<i>n</i>	<i>t</i>	<i>r</i>	<i>e</i>		<i>1</i>	<i>-</i>	<i>9</i>

No se puede utilizar el símbolo ¿ ni el de grados (°)









Loop:

<i>T</i>	<i>x</i>	<i>.</i>	<i>y</i>	<i>c</i>		<i>H</i>	<i>X</i>	<i>X</i>	<i>%</i>		<i>L</i>	<i>x</i>	<i>x</i>	<i>K</i>	
<i>2</i>	<i>3</i>	<i>:</i>	<i>o</i>	<i>1</i>	<i>:</i>	<i>2</i>	<i>3</i>		<i>P</i>	<i>l</i>	<i>a</i>	<i>n</i>	<i>t</i>	<i>a</i>	

Como no se puede poner ° se pone C indicando que son grados Celsius. La humedad solo ocupa 2 valores debido que el 100% seria estar en agua. La luz se mide en Lux y el LDR lo mide en klux de ahí la K de mil.

Creamos las funciones que uniremos al final en el código principal (explicado entero en el apartado 6 de la memoria):

Hay que incluir LiquidCrystal en la biblioteca para que pueda leerlo al poner #include.

	dht_nonblocking.cpp	27/06/2019 11:33	Archivo CPP	9 KB
	dht_nonblocking.h	27/06/2019 11:33	Archivo H	2 KB
	IRremote.cpp	27/06/2019 11:33	Archivo CPP	33 KB
	IRremote.h	27/06/2019 11:33	Archivo H	5 KB
	IRremoteInt.h	27/06/2019 11:33	Archivo H	16 KB
	led_pot_LDR_DHT_Display.ino	01/07/2019 19:08	Arduino file	7 KB
	LiquidCrystal.cpp	27/06/2019 11:33	Archivo CPP	10 KB
	LiquidCrystal.h	27/06/2019 11:33	Archivo H	3 KB

Primero en el setup están todas estas funciones, al estar en el set up solo se ejecutan UNA vez y luego entra en el loop que nunca sale a no ser que se reinicie la placa Arduino. La función que añadimos es la de displayLines (); y la de tipodeplanta;

DisplayLines (); se encuentra en primer lugar porque es la encargada de inicializar el lcd, luego enseñara el mensaje Tipo de planta elegir entre el 1-9 esto lo hace la función tipodeplanta ();

```
void setup() {
  init_led();

  turn_down_led();

  displayLines();

  tipodeplanta();

  moistureSensor();

  //planta=1;
}
```

```

void displayLines() {
  lcd.begin (16, 2);
}

//Funciones para el mando
void tipodeplanta() {
  Serial.println("¿ Tipo de planta ( 1-9 )?");

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Tipo de planta?");
  lcd.setCursor(0, 1);
  lcd.print("Elegir entre 1-9");

  irrecv.enableIRIn();
  //planta=1;
}

```

Ahora vamos al loop

```

void loop() {

  if (planta == 0) {

    IR_Read();

  } else {

    // Read photo resistor LUX
    LDR_Read();
    //
    moisture_Read();
    // Read temp + humedad
    DHT_Read();
    //
    displayRead();
    //
    // Enciende Led
    if (valorLDR > 75) {
      turn_down_led();
    } else {
      turn_up_led();
    }
  }
}

```

*If (valorLDR > 75) que apague el led es un comando de prueba.*

En el displayRead (); esta las variables que quiero enseñar de humedad, temperatura y el del LDR, añadiendo un reloj. Se decidió poner un reloj para contar los minutos que lleva encendido y ser más fácil a la hora de testearlo.

1. Redondear las variables de temperatura, humedad y luz.



```

void displayRead() {
    float t1 = temperature;
    float t2 = Humidity2;
    float t3 = valorLDR;

    int temperature = (int) t1;
    int Humidity2 = (int) t2;
    int valorLDR = (int) t3;

```

lcd.setCursor-> empieza a escribir

Lcd.print -> Escribe en el display.

```

lcd.setCursor(0, 0);
lcd.print("T=" + String(temperature) + " " + "H=" + String(Humidity2) + " " + "L=" + String(valorLDR) + " ");

```

2. Reloj. El comando final esta sacado de internet. El que se intentó programar desde la base es el siguiente:

```

void loop() {

    lcd.clear();
    if(actualHour<10){

        lcd.setCursor(0, 0);
        lcd.print(0);
        lcd.setCursor(1,0);
        lcd.print(actualHour);
    }
    else{
        lcd.setCursor(0,0);
        lcd.print(actualHour);
    }

    lcd.setCursor(2, 0);
    lcd.print(":");

    if(actualMin< 10){
        lcd.setCursor(3,0);
        lcd.print(0);
        lcd.setCursor(4,0);
        lcd.print(actualMin);
    }
    else{
        lcd.setCursor(3, 0);
        lcd.print(actualMin);
    }
}

```

```

lcd.setCursor(5, 0);
lcd.print(":");

if(actualSec< 10){
    lcd.setCursor(6,0);
    lcd.print(0);
    lcd.setCursor(7,0);
    lcd.print(actualSec);
}
else{
    lcd.setCursor(6,0);
    lcd.print(actualSec);
}

if(!setTimeButton1 && !setTimeButton2 && !setTimeButton3){
    actualSec++;
    if(actualSec == 60){
        actualMin++;
        actualSec = 0;
    }
    if (actualMin == 60){
        actualHour++;
        actualMin = 0;
    }
}

Serial.print(actualHour);
Serial.print(":");
Serial.print(actualMin);

```

El código funcionaba correctamente por un tiempo, pero luego se descomponía en tiempos diferentes. Debido a no saber dónde estaba el falló se implementó un nuevo código fiable de internet (código en la bibliografía):

```

int days;
int hours;
int minutes;
int seconds;
int timeElapsed;
long timeNow;

// Constants
long day = 86400000; // 86400000 milliseconds in a day
long hour = 3600000; // 3600000 milliseconds in an hour
long minute = 60000; // 60000 milliseconds in a minute
long second = 1000; // 1000 milliseconds in a second

```

```

timeNow = millis() - previousMillis;

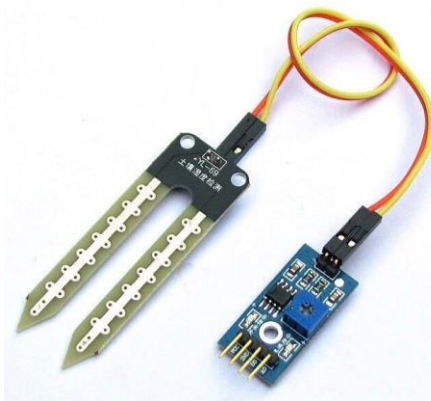
days = timeNow / day ;                               //number of days
hours = (timeNow % day) / hour;                       //the remainder from days division (in milliseconds) divided by hours, this gives the full hours
minutes = ((timeNow % day) % hour) / minute ;        //and so on...
seconds = (((timeNow % day) % hour) % minute) / second;

if (hours>=24) {
  previousMillis = millis();
  timeNow = millis() - previousMillis;
  days = timeNow / day ;                               //number of days
  hours = (timeNow % day) / hour;                       //the remainder from days division (in milliseconds) divided by hours, this gives the full hours
  minutes = ((timeNow % day) % hour) / minute ;        //and so on...
  seconds = (((timeNow % day) % hour) % minute) / second;
}

lcd.setCursor(0, 1);
lcd.print( (hours < 10 ? "0:" : "") + String(hours) + ":" + (minutes < 10 ? "0:" : "") + String(minutes) + ":" + (seconds < 10 ? "0:" : "") + String (seconds) + "
");
}

```

## 5.7\_Modulo 7: Humedad de Tierra.



Sensor de humedad del suelo o sensor yl-69. Las conexiones son las siguientes:

VCC: Tensión de alimentación a 5V

GND: A tierra

Ao: Salida analógica que entrega una tensión proporcional a la humedad. Esta en A2

Como está escrito en la metodología, primero comprobamos que funciona

Fig. 12 – Sensor yl-69.



Se mete el sensor en una maceta con tierra y para humedecer la tierra mojamos la tierra, no toque directamente al sensor, aunque este pueda soportar estar en agua completamente. Para que nos den valores secos simplemente sacamos el sensor.

El código de prueba es el siguiente:

Si el sensorValue (que son los valores que le entran al sensor) depende de las comparaciones se averiguara si está en tierra, seco, húmedo o en agua. El resultado saldrá en el monitor serie.

```
void setup()
{
  Serial.begin(9600);
  pinMode(A2, INPUT);
}

void loop()
{
  int SensorValue = analogRead(A2); //esta en el analógico
  //Humidity2 =map(SensorValue,0,1023,0,100); mapeo para más adelante
  Serial.print(SensorValue); Serial.print(" - ");

  if(SensorValue >= 1000) {
    Serial.println("Sensor is not in the Soil or DISCONNECTED");
  }
  if(SensorValue < 1000 && SensorValue >= 600) {
    Serial.println("Soil is DRY");
  }
  if(SensorValue < 600 && SensorValue >= 370) {
    Serial.println("Soil is HUMID");
  }
  if(SensorValue < 370) {
    Serial.println("Sensor in WATER");
  }
  delay(50);
}
```

A continuación, se explica las funciones que se añadirán al código final:

Empezando con las definiciones de donde esta y su variable (las marcadas en amarillo).

```
// definiciones
int ledPin = 6;
int PotentPin = A0;
int LDRPin = A1;
int ledMoisture=A2;

// Valor de la Luz 0..100%
int valorLDR;
// definición del ir
int receiver = 3;
IRrecv irrecv(receiver);
decode_results results;
LiquidCrystal lcd(7, 8, 9, 10, 5, 12);
int planta = 0;

// Valores globales a display
int ValLux = 0;
float temperature;
float humidity;
int ValPot = 0;
int SensorValue;
int Humidity2;
int previousMillis;
```

En el SetUp se añadirá la función al final denominada `moistureSensor()`; la cual inicializa el sensor del suelo:

```
void moistureSensor() {  
  pinMode(ledMoisture, INPUT);  
}
```

En el loop es donde ocurre todo el proceso con la función `moisture_Read()`; Como tiene valores analógicos que van del 0-1023 hacemos un mapeo cambiándolo a porcentaje siendo 100% estar en agua y 0% en seco/ sin tierra. También hay que añadir que el valor `Humidity2` salga en el display, para ello vamos al comando `displayRead()`; del loop(última imagen):

```
void moisture_Read() {  
  int SensorValue = analogRead(ledMoisture);  
  
  Humidity2 = map(SensorValue, 0, 1023, 100, 0);  
  
}
```

#### MAPEO DE LA HUMEDAD

```
lcd.setCursor(0, 0);  
lcd.print("T=" + String(temperature) + " " + "H=" + String(Humidity2) + " " + "L=" + String(valorLDR) + " ");
```

#### SALGA LA HUMEDAD EN EL LCD

## 5.8\_Modulo 8: Thermoresistor.

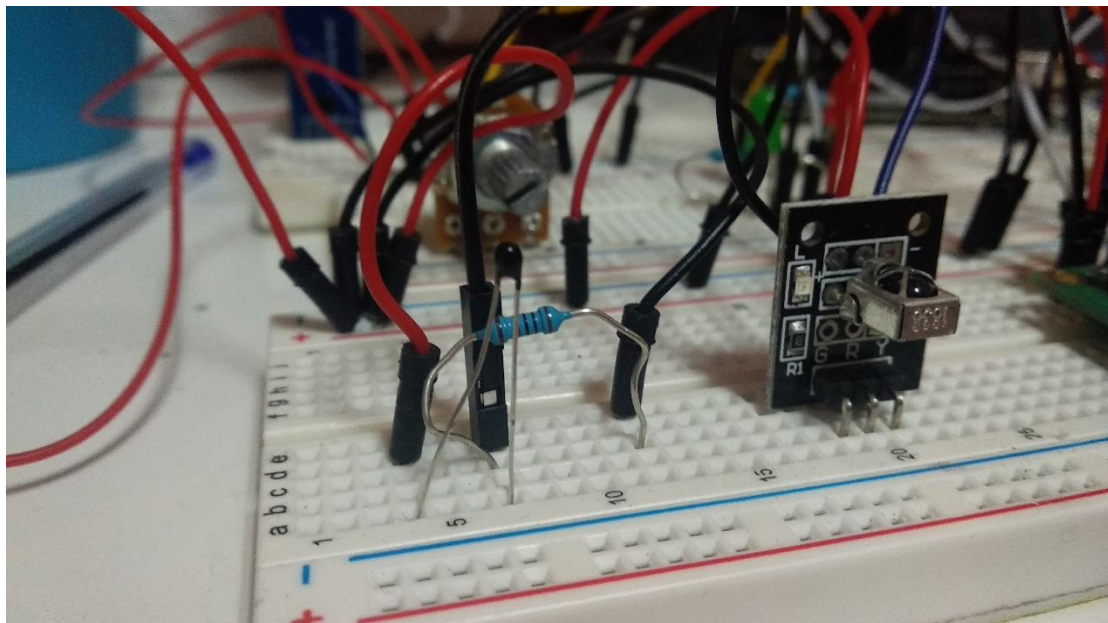


Fig. 13 – thermoresistor.

El DHT no media bien la temperatura, se cree que es por haberlo forzado la entrada en el protoboard que ha creado un tipo de fisura.

No había tiempo para comprar otra pieza así que se ingenió con un thermoresistor que tenía.

Utilizando la formula Steinhart-Hart que dice así:

$$\frac{1}{T} = A + B * \ln(R) + C * [\ln(R)]^3$$

- T es la temperatura en kelvis



- R es la resistencia de T en ohmios

- A, B y C son los coeficientes de Steinhart-Hart que dependen del tipo y el modelo de thermoresistor usado. Para saber cuál se accedió a una página en la cual venían todos los valores de varios modelos (para saber cuál era cada uno tenía diferentes funciones, cabezas diferentes y patas distintas), resultando ser el nuestro el MF 52 NTC thermoresistor. Probamos en el código para asegurarse, lo comparamos los resultados con una estación de meteorología. Dando lugar: A = 0.001129148, B = 0.000234125, C = 0.0000000876741.

Fig. 14 – Estación meteorológica.

Con la ecuación la despejamos y pasamos los kelvin en grados, pero primero podemos las variables.

```
//definiciones del termoresistor
int ThermistorPin = A3;
int Vo;
float R1 = 10000;
float logR2, R2, T;
float A = 0.001129148, B = 0.000234125, C = 0.0000000876741;
```

En el loop añadimos la función Temp\_Read (); que dice así:

```

void Temp_Read() {
  Vo=analogRead(ThermistorPin);
  R2=R1 * (1024 / (float)Vo - 1.0);
  logR2 = log(R2);
  T = (1.0 / (A + B*logR2 + C*logR2*logR2*logR2));
  T = T - 273.15;
  //T = (T * 9.0) / 5.0 + 32.0; //<- esto es para farenheit
  //T = map(Vo, -55, 125201, 0, 100);

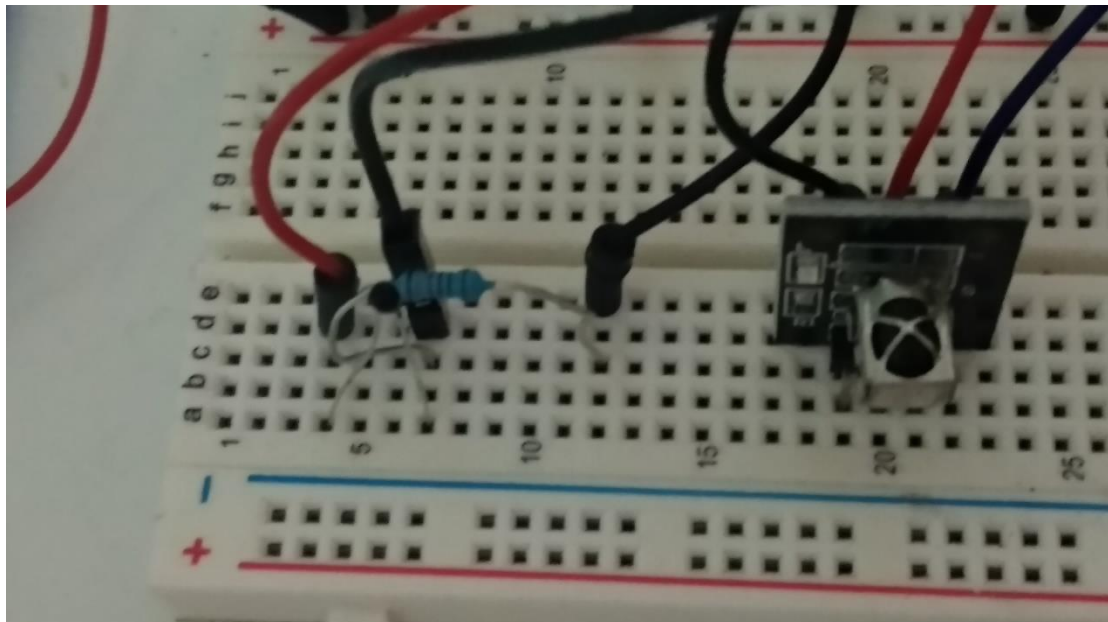
  Serial.print("Temperature: ");
  Serial.print(T);
  Serial.println(" °C");

  //delay(500);
}

```

Y cambiamos en el display donde antes ponía temperature por T.

La conexión del theresistor es:



De izquierda a derecha:

El cable rojo va a 5voltios es la alimentación para que funcione. En la otra pata del thermoresistor Delante esta una resistencia de 1k y siguiendo delante esta un cable negro (por que faltaban de color) que va al analógico 3 y el de la derecha del todo va a tierra.

### 5.9\_Modulo 9: Led de alarma.



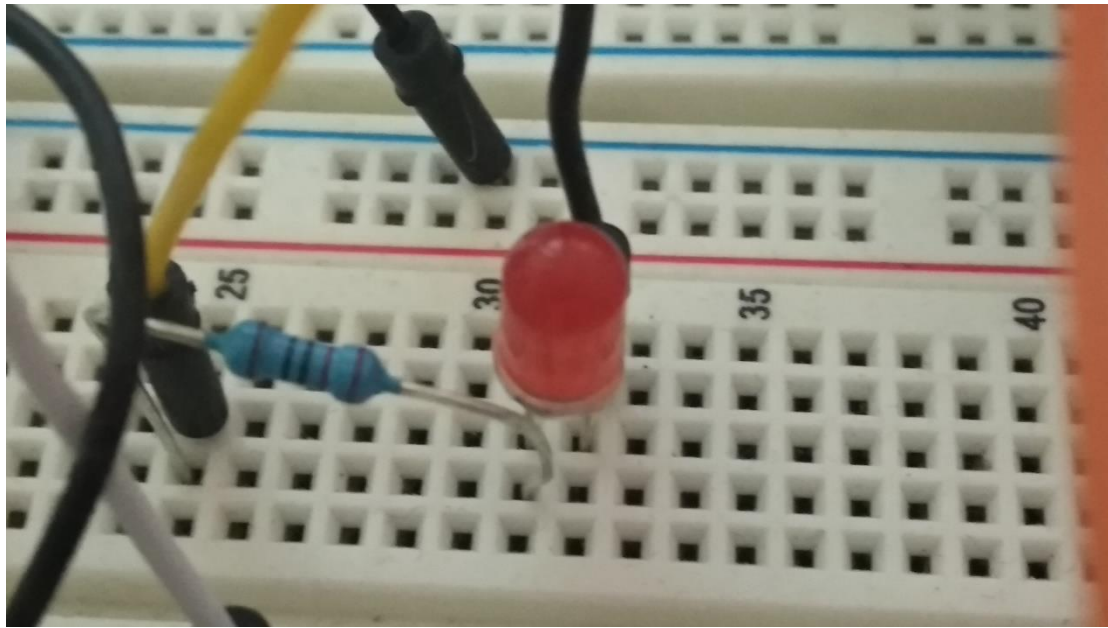


Fig. 15 – Led de alarma y su resistencia de 330 ohmios.

La conexión es la siguiente el cátodo va a ground es decir la pata más corta que tiene enfrente el cable negro va hacia tierra y la resistencia que está detrás del ánodo va al 13 digital.

La alarma al principio se quería introducir para que dé señal cuando pasa cierto tiempo y aún no se ha regado. También cuando la planta esté en temperaturas críticas para ella y en lux no aconsejable.

Al final tan solo se mantuvo poner alarma cuando la temperatura es mínima y máxima, que puede llegar a ser peligrosa para la planta.

```
int flagAlarm=false;
int flagWatering=false;

int AlarmMinTemp[iNumArray-1]={0,-10,2,13,-3,9,0,10,0};
//int AlarmMaxLux[iNumArray-1]={71,91,70,80,70,70,60,70,70};
int AlarmMaxTemp[iNumArray-1]={41,46,40,45,46,30,45,45,30};
```

Los valores elegidos se verán en el próximo modulo.



```

//Función del led de emergencia
void turn_on_emergency() {
  if (flagAlarm) {
    digitalWrite(ledAlarma, HIGH);
  }else {
    digitalWrite(ledAlarma, LOW);
  }
}

// Funciones del led, pin 3

void init_led() {
  //
  pinMode(ledAlarma, OUTPUT);
  pinMode(ledPin, OUTPUT);
  //
  Serial.begin(9600);
}

flagAlarm= false; //No hay alarma
flagAlarm= (T < AlarmMinTemp[planta-1]? true : flagAlarm);
flagAlarm= (T > AlarmMaxTemp[planta-1]? true : flagAlarm);

```

Aconsejable Si la temperatura llega a ser igual/ mayor o menor que la temperatura en la que puede morir una planta, este dará alarma y además saldrá en el led.

```

//Alarma, Planta y regando
lcd.setCursor(8, 1);
if (flagAlarm) {
  lcd.print(" ALARMA!");
  lcd.blink();
} else {
  lcd.noBlink();
  if (flagWatering == true) {
    lcd.print(" Regando");
  } else {
    lcd.print(" " + sPlantas[planta-1] + " ");
  }
}
}

```

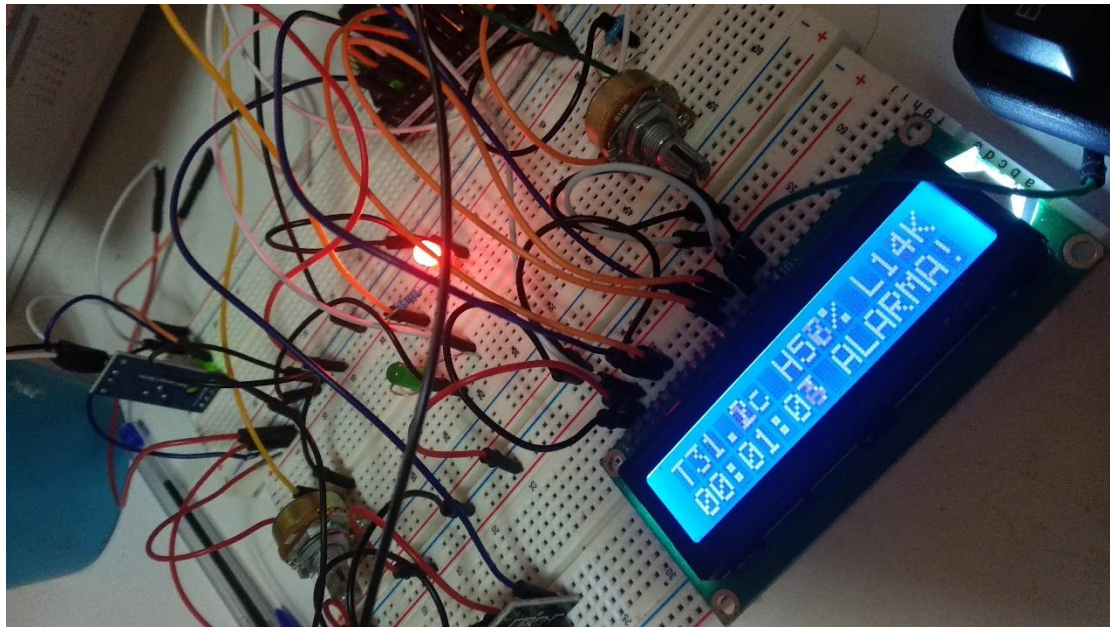


Fig. 16- Led de alarma prueba.

### *5.10\_Modulo 10: Investigación de plantas.*

Se eligieron plantas al azar y por la información que se podría encontrar en las universidades, quedando al final: Rosa, Olivo, Tomate, Algodón, Naranja, Arroz, Patatas, Níspero y al final un Ejemplo para poder comprobar todo el sistema.

El siguiente paso es conseguir la información de cada planta cuando es más optimo regar. Buscando en todo tipo de fuentes como licenciadas como de artículos, no se pudo obtener estos valores. Aun intentando aprovechando el uso de las redes sociales (en este caso usé Facebook), me uní a varios grupos de ingenieros agrónomos y pregunté tanto como universidades como a empresas privadas.

Los grupos con los que probé fueron:



Fig. 17– Grupos de Facebook para conseguir información.

## Las universidades con las que contacte



Fig. 18– Preguntas a las universidades por información.

Una universidad denominada Colegio Oficial de Ingenieros Agrónomos de Centro y Canarias, me dio el email al cual me contesto que aún no hay ningún estudio hecho con esos datos y que además cada planta es un ser vivo y se comportan diferente a pesar de ser de la misma especie.

Al perder la parte biológica del proyecto se tuvo que pasar al plan B que es hacer una estimación.

Con la información proporcionada de infoagro, se estudia a que temperatura la planta es cuando está en el máximo de trabajo, es decir utiliza todos los nutrientes proporcionados. Dando estos valores

Empezando desde la derecha cada valor son de: Rosa, olivo, tomate, algodón, naranja, arroz, patata, níspero, el ejemplo.

```
int MinTemp[iNumArray-1]={14,-10,13,14,5,10,13,15,0};
int MaxTemp[iNumArray-1]={25,43,26,30,37,40,18,37,30};
```

También el estudio en el cual la planta puede fallecer si se llegan a ciertas temperaturas:

```
int AlarmMinTemp[iNumArray-1]={0,-10,2,13,-3,9,0,10,0};
//int AlarmMaxLux[iNumArray-1]={71,91,70,80,70,70,60,70,70};
int AlarmMaxTemp[iNumArray-1]={41,46,40,45,46,30,45,45,30};
```

Luego para los lux vemos los lux que hay en 24 horas y comparamos los lux que es más favorable para regar, una tabla hecha por Jim Palmer:

Iluminancia	Abr.	Ejemplo
0,00005 lux	50 µlx	Luz de una estrella (Vista desde la tierra)
0,0001 lux	100 µlx	Cielo nocturno nublado, luna nueva
0,001 lux	1 mlx	Cielo nocturno despejado, luna nueva
0,01 lux	10 mlx	Cielo nocturno despejado, cuarto creciente o menguante
0,25 lux	250 mlx	Luna llena en una noche despejada <sup>1</sup>
1 lux	1 lx	Luna llena a gran altitud en latitudes tropicales <sup>2</sup>
3 lux	3 lx	Límite oscuro del crepúsculo bajo un cielo despejado <sup>3</sup>
100 lux	1 hlx	Pasillo en una zona de paso
300 lux	3 hlx	Sala de reuniones
500 lux	5 hlx	Oficina bien iluminada
600 lux	6 hlx	Salida o puesta de sol en un día despejado.
1000 lux	1 klx	Iluminación habitual en un estudio de televisión
32.000 lux	32 klx	Luz solar en un día medio (mín.)
100.000 lux	100 klx	Luz solar en un día medio (máx.)

El código con los valores, mismo orden de planta que con la temperatura:

```
int MinLux[iNumArray-1]={5,5,5,5,5,5,5,5,5};
int MaxLux[iNumArray-1]={70,90,65,65,73,65,70,64,30};
```

Por último, la temperatura, si la temperatura es menor a la establecida este regara si todos los demás parámetros concuerdan (mismo orden que la temperatura):

```
int MaxHum[iNumArray-1]={45,30,50,50,45,40,50,58,40};
```

Todos estos valores han sido sacados de infoagro y de otras fuentes, pero siguen siendo unas aproximaciones y no valores óptimos.

## 6. Resultados obtenidos:

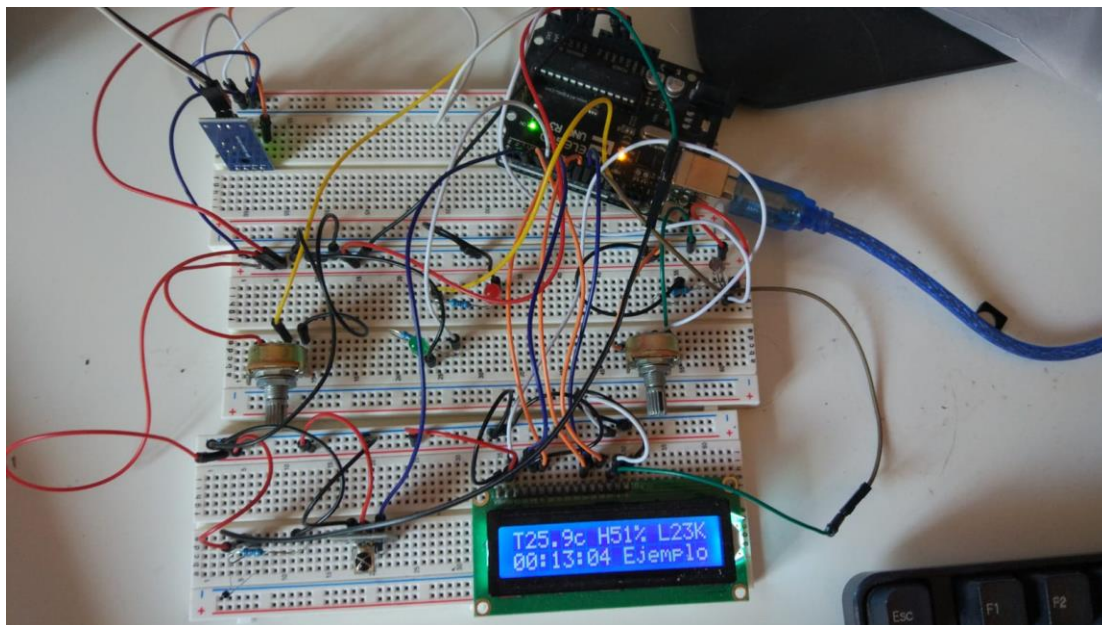


Fig. 19– Riego inteligente Final.

El programa consta de los siguientes módulos (el programa completo se encuentra en los anexos):

- **Librería.**
  - Pantalla LCD (LiquidCrystal.h)
  - Infrarrojos (IRremote.h)
- **Variable.**
  - Definiciones de los pines analógicos y digitales (LCD).
  - Lecturas analógicas: Humedad, Temperatura, Luz
  - Reloj tiempo real (HH:MM:SS): Definición de día, hora minutos y segundos.
  - Tipos de plantas (máxima 9): las nueve plantas elegidas son Rosa, Olivo, Tomate, Algodón, Naranja, Arroz, Patata, Níspero y un ejemplo para probar el código.
  - Valores deseados para regar según la planta elegida (máx. 9)
    - Temperatura máxima y mínima.

- Luz máxima y mínima.
- Humedad máxima. Solo es humedad máxima debido a que solo nos indica si ha regado o no.
- Alarmas temperatura (máxima y mínima). Debido a que las plantas presentan temperaturas que no pueden soportar.
- Variables de alarmas.
- **Funciones.**
  - Filtrajes de las lecturas analógicas (Filtro\_Medicion): Retorna la media aritméticas de las 10 últimas mediciones. Coge las 10 últimas lecturas de los sensores y el último valor obtenido lo guarda para la siguiente media. Así consigue bajar el margen de error.
  - Alarmas: Enciende o apaga led alarma y el de riego.
  - Inicialización de los periféricos: led, puertos, periféricos (LCD, IR, Humedad).
  - Lecturas analógicas: Temperatura, Humedad, Luz.
  - Funciones de escribir LCD.
  - Leer IR.
  - Calcular la hora en tiempo real. Esto se ha decidido dejarlo para en un futuro desarrollarlo.
- **Configuración/Setup.**
  - Inicializas los puertos (Leds, LCD, Sensores de Humedad, Luz y temperatura) en este orden:
    - Empieza inicializando el led.
    - Display.
    - Pregunta en el display el tipo de planta que lo recibe la respuesta los infrarrojos.
    - Inicia el sensor de humedad de tierra.
- **Bucle.**
  - Lee el tipo de planta.
  - Riego inteligente en sí:
    - Lee las 3 variables esenciales: temperatura, humedad, luz.

- Filtra para evitar ruidos.
- La humedad es comparada para ver si se riega o no.
- Se analiza temperatura y humedad si es correcto para regar.
- Control de alarma: Caso exceda temperatura por alta o baja, que sea no beneficiosa la salud de la planta, ¡aparece en el display ALARMA! y se enciende el led rojo de emergencia. Se apagará cuando los valores entrantes sean en la zona segura.

## 7. Conclusiones, trabajo por hacer y mejoras:

El objetivo principal de crear un sistema de control si ha sido cumplido y también el sistema puede regar en cualquier zona geográfica, el ahorro de agua no es algo realmente logrado debido a que no hay un estudio científico comprobado que aporte un óptimo uso del agua usada en el riego. Los objetivos secundarios también han sido realizados, se ha explorado ambos lenguajes de Arduino y de C+ además de aprender el montaje de todos los elementos. Aunque uno de los objetivos secundarios es la información, se ha logrado aprender a buscar soluciones e intentar encontrar nuevas fuentes en las redes sociales.

Mejoras en el proyecto hay varias, la principal es conseguir la base científica que hace que el proyecto esté realmente acabado. La segunda es aporta otro elemento el cual busque la información en una página web en internet y la tercera mejora pensada es en vez de que la alarma sea un led, que por bluetooth llegué un mensaje al móvil avisando que le pasa a la planta y si pregunta el consumidor por la planta, que el sistema responda con los parámetros obtenidos.

## 8. Documentación y Bibliografía:

ARDUINO

[www.eleegoo.com](http://www.eleegoo.com)

[www.arduino.com](http://www.arduino.com)

[\(199\) \[Test Run\]Arduino Uno LCD 16x2 Scroll text, Both line text, Blink text ,Single row text - YouTube](#)

[Arduino - LiquidCrystalTextDirection](#)

[Arduino LCD 16x2 Both Line Text](#)

[Contador de tiempo | MiArduino](#)

[Ayuda con temporizador con LCD 16 x 2.](#)

[formula para calcular el resto de una division - Buscar con Google](#)

[A function definition is not allowed here before '{' token - Help - The freeCodeCamp Forum](#)

[Arduino: How do you reset millis\(\) ? - Bald Engineer](#)

[programming - Resetting millis\(\) and micros\(\) - Arduino Stack Exchange](#)

[Ayuda con la funcion MAP](#)

[Funciones | Aprendiendo Arduino](#)

[Arduino/Moisture\\_Sensor.ino at master · TasmanianDevilYouTube/Arduino · GitHub](#)

[\(208\) Reloj digital con Arduino + LCD \[Muy facil\] - YouTube](#)

[Tutorial para realizar un sencillo y funcional Cronometro](#)

[\(208\) Programando con Arduino | 9 - Reloj digital con ARDUINO UNO + LCD - YouTube](#)

[Temporizador: Arduino + LCD](#)

[Como medir el tiempo con Arduino y la librería Time. | Leantec.ES](#)

[Arduino Temporizador.](#)

[termoresistor - Buscar con Google](#)

[Focusens Tech - Thermal Sensitive Components, Temperature / Humidity Transducer](#)

[OpenSmart](#)

[Make an Arduino Temperature Sensor \(Thermistor Tutorial\)](#)

[Arduino en español: const](#)

[Arduino en español: int](#)

INFORMACIÓN DE PLANTAS

[AGRARIA 20 PAG29-47.pdf](#)

[Cálculo del balance energético de una plantación de Populus deltoides clon Lux con fines energéticos en un sitio con ambiente mediterráneo](#)

[untitled](#)



[Manejo riego goteo olivo\\_CIEBV2006.pdf](#)

[Monitorización del clima, planta y suelos para optimización de riego en cultivo de olivo para minimizar el consumo de agua en Cidamón](#)

[Agricultura. El cultivo de la pera. 1ª parte.](#)

[Plantación de olivo en superintensivo](#)

[heladas olivar andaluz tecnicas teledeteccion.pdf](#)

[02.pdf](#)

[El cultivo de la rosa](#)

[Agroinformación - El cultivo del tomate. 1ª parte.](#)

[La clase String](#)

[Tomato Production](#)

[206832](#)

[【 Riego por Goteo para Tomates 】 | ¿Cuánta agua necesitan? Todo aquí!](#)

[Crop Guide: Growing Olives - Haifa Group](#)

[Influencia del Clima en los Olivos \(lluvia, sequía, enfermedades...\)](#)

[PROGRAMACIÓN DEL RIEGO DE OLIVAR EN ANDALUCÍA](#)

[www.facebook.com](#)

## 9. Anexos

Todo el código final:



```
#include "IRremote.h"
#include "LiquidCrystal.h"

//definiciones del termoresistor
int ThermistorPin = A3;
int Vo;
float R1 = 10000; //pull up resistor = resistencia que tira del thermoresistor
float logR2, R2, T;
float A = 0.001129148, B = 0.000234125, C = 0.0000000876741;
//float A = 1.009249522e-03, B = 2.378405444e-04, C = 2.019202697e-07;

// definiciones del led
int ledPin = 6;
int PotentPin = A0;
int LDRPin = A1;
int ledMoisture=A2;

//definiciones del led de alarma
#define ledAlarma 13

// definición del ir
int receiver = 3; //reciber pin = pin
IRrecv irrecv(receiver);
decode_results results;
//Pines del LCD
LiquidCrystal lcd(7, 8, 9, 10, 5, 12);

// Valores globales a display
// Valor de la Luz 0..100%
int valorLDR;
int ValPot = 0;
int SensorValue;
int Humidity;
int previousMillis;

// definiciones del reloj
int days;
int hours;
int minutes;
int seconds;

//int Algodón, Olivo, Tomate, Peral, Rosal, Arce Japones, Nispero, Cactus, Orquidea
int planta = 0;
String sPlantas[9]={"Rosa", "Olivo", "Tomate", "Algodon", "Naranja", "Arroz", "Patatas", "Nispero", "Ejemplo" };

const int iNumArray=10; //Cuantas mediciones para hacer la meria aritmetica
//
int ValTemp[iNumArray]={0,0,0,0,0,0,0,0,0,0}; //Los zeros son para que se inicialize, si no los pongo da una media aritmetica falsa.
int FlagFirstTemp=true;
//
int ValHum[iNumArray]={0,0,0,0,0,0,0,0,0,0};
int FlagFirstHum=true;
//
int ValLux[iNumArray]={0,0,0,0,0,0,0,0,0,0};
int FlagFirstLux=true;
//
```

```

int flagAlarm=false;
int flagWatering=false;

//
//
int MinTemp[iNumArray-1]={14,-10,13,14,5,10,13,15,0};
int MaxTemp[iNumArray-1]={25,43,26,30,37,40,18,37,30};
//
int MaxHum[iNumArray-1]={45,30,50,50,45,40,50,58,40};
//
int MinLux[iNumArray-1]={5,5,5,5,5,5,5,5,5};
int MaxLux[iNumArray-1]={70,90,65,65,73,65,70,64,30};
//
int AlarmMinTemp[iNumArray-1]={0,-10,2,13,-3,9,0,10,0};
//int AlarmMaxLux[iNumArray-1]={71,91,70,80,70,70,60,70,70};
int AlarmMaxTemp[iNumArray-1]={41,46,40,45,46,30,45,45,30};

// Average functions
int Filtra_Medicion(int iValue, int *pArrayValue, int *pFlagFirst) {
    int iLoop;
    int Average = iValue;
    //
    if (*pFlagFirst) {
        for (iLoop = 0; iLoop<=iNumArray-1; iLoop++) {
            *(pArrayValue + iLoop) = iValue;
        }

        *pFlagFirst = false;
    }
    //
    for (iLoop = iNumArray-2; iLoop>=0; iLoop--) {
        *(pArrayValue + 1 + iLoop) = *(pArrayValue + iLoop);
    }
    *(pArrayValue + 0) = iValue;
    //
    Average = 0;
    for (iLoop = 0; iLoop<=iNumArray-1; iLoop++) {
        Average += *(pArrayValue + iLoop);
    }
    Average = (int) round(Average / iNumArray);
    return(Average);
}

//Función del led de emergencia
void turn_on_emergency() {
    if (flagAlarm) {
        digitalWrite(ledAlarma, HIGH);
    }else {
        digitalWrite(ledAlarma, LOW);
    }
}

// Funciones del led, pin 3

```

```

void init_led() {
    //
    pinMode(ledAlarma, OUTPUT);
    pinMode(ledPin, OUTPUT);
    //
    Serial.begin(9600);
}
void turn_up_led() {
    //
    int sensorValue = 0;
    // 0 .. 1023
    sensorValue = analogRead(PotentPin);
    // 0 .. 255
    ValPot = map(sensorValue, 0, 1023, 0, 255);

    analogWrite(ledPin, ValPot);
    //analogWrite(ledPin,255);

    Serial.println("");
    Serial.print(">>>>>>> Pot sensorValue=");
    Serial.print(sensorValue);
    Serial.print(" ValPot =");
    Serial.print(ValPot);
    Serial.println(" <<<<<<<<");
}
void turn_down_led() {
    analogWrite(ledPin, 0);
}

//Funcion del sensor de luz
//100 klx - 100.000 lux máx de unidad de medida de la luz
//
void LDR_Read() {
    int fade;
    fade = analogRead(LDRPin);

    Serial.println("");
    Serial.print("before Lux =");
    Serial.println(fade);

    // Filtra medicion
    fade=Filtra_Medicion(fade, &ValLux[0], &FlagFirstLux);

    valorLDR = map(fade, 0, 1023, 0, 100);

    Serial.print("Lux fade filtered=");
    Serial.print(fade);
    Serial.print(" valorLDR =");
    Serial.println(valorLDR);
}

void Temp_Read() {
    Vo=analogRead(ThermistorPin);

```

```

Serial.println("");
Serial.print("before Vo =");
Serial.println(Vo);

// Filtra medicion
Vo=Filtro_Medicion(Vo, &ValTemp[0], &FlagFirstTemp);

R2=R1 * (1024 /((float)Vo - 1.0);
logR2 = log(R2);
T = (1.0 / (A + B*logR2 + C*logR2*logR2*logR2));
T = T - 273.15;
//
// Reajuste.....
//T = T - 4;
//

Serial.println("");
Serial.print("filtered Vo =");
Serial.println(Vo);

Serial.print("Temperature: ");
Serial.print(T);
Serial.print(" °C");
Serial.println("");

//delay(500);
}

void write_message() {

//Serial.print("Planta "); Serial.println(planta);
//Serial.print("Lux "); Serial.println(valorLDR);
//Serial.print("Pot "); Serial.println(ValPot);
//
//Serial.print( "T = " );
//Serial.print( T, 1 );
//Serial.println( "grados" );
//Serial.print( Humidity, 1 );
//Serial.println( "%" );
}

//Funciones para el mando
void tipodeplanta() {
Serial.println("¿ Tipo de planta ( 1-9 )?");

lcd.clear();
lcd.setCursor(0, 0);
lcd.print("Tipo de planta?");
lcd.setCursor(0, 1);
lcd.print("Elegir entre 1-9");

irrecv.enableIRIn();
}

void translateIR() {

```

```

switch (results.value) {
    case 0xFF30CF: Serial.println("1");    planta = 1; break;
    case 0xFF18E7: Serial.println("2");    planta = 2; break;
    case 0xFF7A85: Serial.println("3");    planta = 3; break;
    case 0xFF10EF: Serial.println("4");    planta = 4; break;
    case 0xFF38C7: Serial.println("5");    planta = 5; break;
    case 0xFF5AA5: Serial.println("6");    planta = 6; break;
    case 0xFF42BD: Serial.println("7");    planta = 7; break;
    case 0xFF4AB5: Serial.println("8");    planta = 8; break;
    case 0xFF52AD: Serial.println("9");    planta = 9; break;
    default:
        Serial.println(" Error  ");
}
delay(100);
if (planta > 0) {
    lcd.clear();
    lcd.noCursor();
    lcd.noAutoscroll();

    //empezar a contar una vez elegido la planta:
    previousMillis=millis();
}
}

void IR_Read() {
    if (irrecv.decode(&results)) {
        translateIR();
        irrecv.resume(); // receive the next value
    }
}

void displayLines() {
    lcd.begin (16, 2);
}

void displayRead() {

    //Datos de los sensores
    int temp=(int) round(T);
    lcd.setCursor(0, 0);
    //lcd.print("T=" + String(temp) + " " + "H=" + String(Humidity) + " " + "L=" + String(valorLDR) + " ");
    lcd.print("T" + String(T,1) + "c " + "H" + String(Humidity) + "% " + "L" + String(valorLDR) + "K ");

    //Reloj
    int days;
    int hours;
    int minutes;
    int seconds;
    long timeNow;

    // Constants
    long day = 86400000; // 86400000 milliseconds in a day
    long hour = 3600000; // 3600000 milliseconds in an hour
    long minute = 60000; // 60000 milliseconds in a minute
    long second = 1000; // 1000 milliseconds in a second

    timeNow = millis() - previousMillis;

    days = timeNow / day ; //number of days
    hours = (timeNow % day) / hour; //the remainder from days division (in milliseconds) divided by hours, this gives the full hours
    minutes = ((timeNow % day) % hour) / minute ; //and so on...

```

```

if (hours>=24) {
    previousMillis = millis();
    timeNow = millis() - previousMillis;
    days = timeNow / day ; //number of days
    hours = (timeNow % day) / hour; //the remainder from days division (in milliseconds) divided by hours, this gives the full hours
    minutes = ((timeNow % day) % hour) / minute ; //and so on...
    seconds = (((timeNow % day) % hour) % minute) / second;
}

lcd.setCursor(0, 1);
lcd.print( (hours < 10 ? "0": "") + String(hours) + ":" + (minutes < 10 ? "0": "") + String(minutes) + ":" + (seconds < 10 ? "0": "") + String (seconds) + "
");

//Alarma, Planta y regando
lcd.setCursor(8, 1);
if (flagAlarm) {
    lcd.print(" ALARMA!");
    lcd.blink();
} else {
    lcd.noBlink();
    if (flagWatering == true) {
        lcd.print(" Regando");
    } else {
        lcd.print(" " + sPlantas[planta-1] + "
");
    }
}
}

void moistureSensor() {
    pinMode(ledMoisture, INPUT);
}

void moisture_Read() {
    int SensorValue = analogRead(ledMoisture);

    Serial.print ("before SensorValue =");
    Serial.println(SensorValue);

    // Filtra medicion
    SensorValue=Filtra_Medicion(SensorValue, &ValHum[0], &FlagFirstHum);

    Humidity = map(SensorValue, 0, 1023, 100, 0);

    Serial.println("");
    Serial.print ("***** Humedad filtered SensorValue =");
    Serial.print (SensorValue);
    Serial.print (" % Humidity =");
    Serial.println(Humidity);
    Serial.println("");
}

void setup() {
    init_led();

    turn_down_led();

    displayLines();

    tipodeplanta();

    moistureSensor();
}

```

```

void loop() {

  if (planta == 0) {

    IR_Read();

  } else {

    // Read photo resistor LUX
    LDR_Read();
    //
    moisture_Read();
    //
    Temp_Read();
    //

    //flagAlarm=(valorLDR > 70 ? true : false);
    //flagWatering=(Humidity < 50 ? true : false);
    flagWatering=(Humidity < MaxHum[planta-1] ? true : false);

    if (T < MinTemp[planta-1]) {
      flagWatering=false;
    }
    if (T > MaxTemp[planta-1]) {
      flagWatering=false;
    }

    if (valorLDR < MinLux[planta-1]) {
      flagWatering=false;
    }

  }

  //flagAlarm=(valorLDR > AlarmMaxLux[planta-1]? true : false);

  flagAlarm= false; //No hay alarma
  flagAlarm= (T < AlarmMinTemp[planta-1]? true : flagAlarm);
  flagAlarm= (T > AlarmMaxTemp[planta-1]? true : flagAlarm);

  if (flagWatering) {
    turn_up_led();
  } else {
    turn_down_led();
  }
  //
  turn_on_emergency();
  //
  displayRead();
  //
  // Enciende Led

  delay(100);
}
}

```

*Riego pruebas y estructura:*





Fig. 20- Al iniciar el sistema primero pregunta el tipo de planta.



Fig. 21- Elegir el tipo de planta con el mando y el IR (lo de allado del display) leerá la señal.

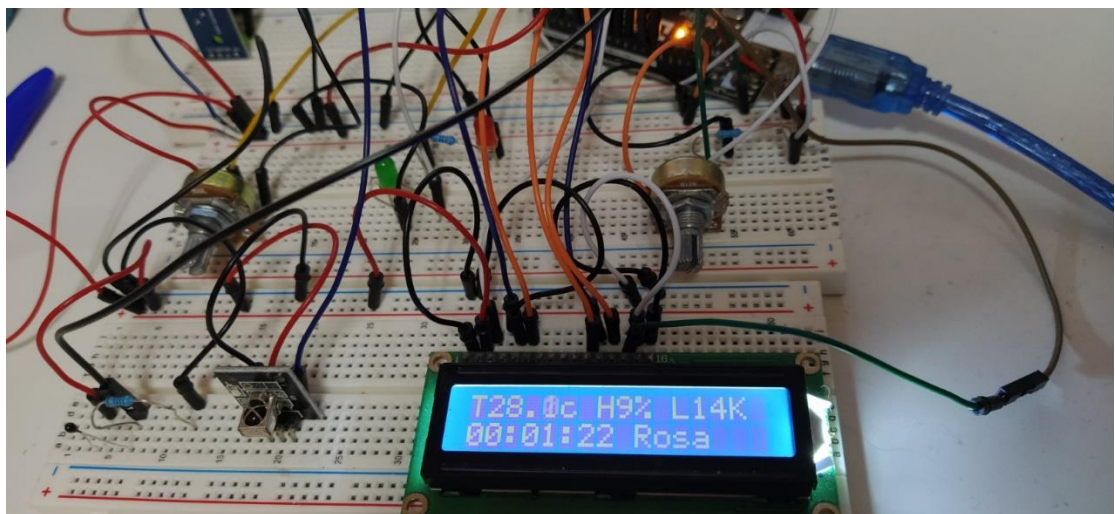


Fig. 22- Eligiendo con el mando la planta Rosa empieza a contary a salir los valores.



Fig. 23-Sensor de humedad de la tierra.



Fig. 24- Para subir el porcentaje de humedad se echa agua del grifo a la tierra.



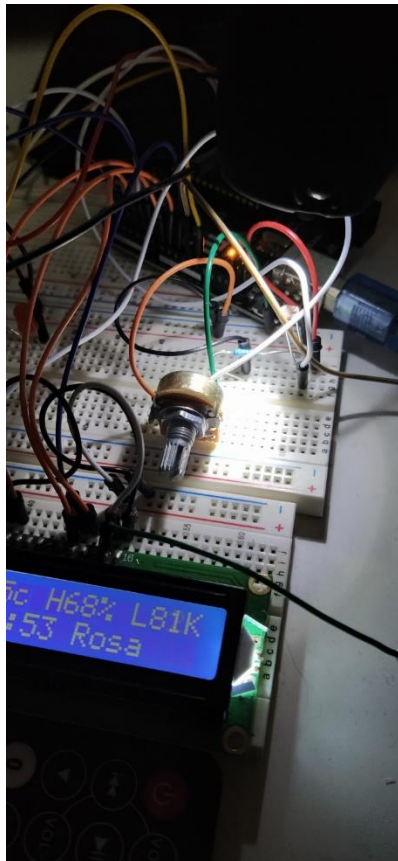


Fig. 25- Para subir los lux utilizamos una linterna.

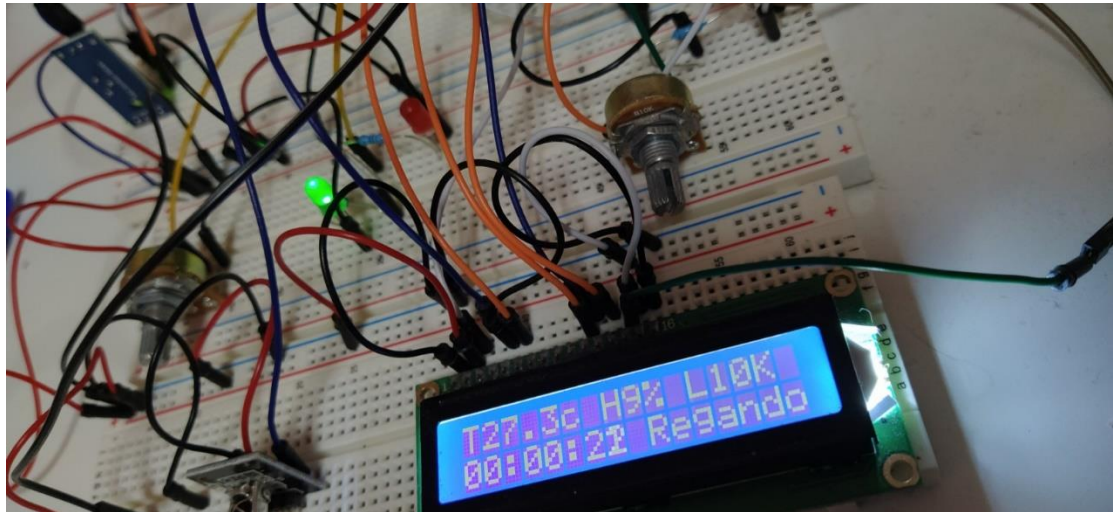


Fig. 26- Con los datos elegidos de planta de ejemplo el sistema ha decidido regar.